# 证　　明

本证明之附件是向本局提交的下列专利申请副本

申　请　日：2003 04 11

申　请　号：03 1 18918.0

申请类别：发明

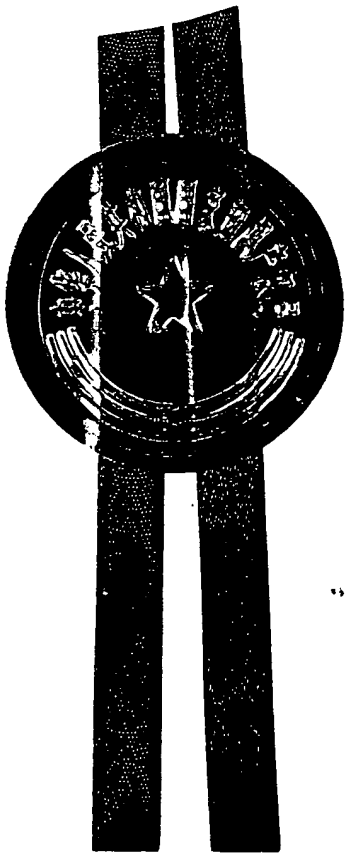
发明创造名称：一种数据包的调度方法



申　请　人：华中科技大学

发明人或设计人：杨宗凯；刘彦；王玉明；范兵





中华人民共和国
国家知识产权局局长 王景川

2004 年 3 月 22 日

# 权 利 要 求 书

1. 一种数据包的调度方法，该方法将待调度的数据包分成新到流队列和积压流队列两部分，由调度器进行调度，其调度步骤为：

(1) 初始化调度节点，设定系统虚拟时间的初始值；

(2) 当某个数据包到达调度节点时，检查这个数据包是否为其所属数据流的第一个数据包，如果是，根据其速率和/或长度挂到新到流队列部分相应的 $Q_1$ 队列的尾部，并通过公式(3)，计算这个数据包的虚起始服务时间，该虚起始服务时间即该数据流的虚起始服务时间；如果这个数据包不是这个数据流的第一个数据包，直接将这个数据包挂在所属数据流的尾部；

(3) 进行调度时，调度器会扫描所有队列中第一个数据流的头数据包的虚起始服务时间，从中找出虚起始服务时间小于系统虚拟时间的合法数据包，再通过公式(4)，计算这些合法数据包的虚结束服务时间，找出具有最小虚结束服务时间的数据包进行发送；

(4) 对选定的头数据包进行发送的过程为：首先将该数据包从所属的数据流 F 中取下来发送，数据流 F 再根据其新的头数据包的速率和长度，挂在积压流队列部分相应的 $Q_2(R_x, L_y)$ 队列的尾部，通过公式(2)更新数据流 F 的虚起始服务时间，即数据流 F 新的头数据包的虚起始服务时间，并通过公式(1)更新系统虚拟时间；

$$V(t+\tau)=\max\left(V(t)+\tau,\min_{i\in B(t)}\left(S_i^{h_i(t)}\right)\right)\cdots\cdots(1)$$

$$S_i^k=\begin{cases}F_i^{k-1}\rightarrow Q_i\neq 0\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots(2)\\ \max\left(V\left(a_i^k\right),F_i^{k-1}\right)\rightarrow Q_i=0\cdots\cdots\cdots\cdots(3)\end{cases}$$

$$F_i^k=S_i^k+\frac{L_i^k}{R_i(t)}\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots(4)$$

(5) 重复步骤(2)至(4)，直至工作完成。

# 说 明 书

## 一种数据包的调度方法

### 技术领域

本发明属于网络通信系统与应用技术领域，具体涉及一种数据包的调度方法，它尤其适用于路由器中数据包的调度。

### 背景技术

随着 Internet 的不断增长，多媒体业务也应用得越来越广泛。由于多媒体业务如语音、图象对带宽和时延要求很高，这就需要路由器具有高效和快速的数据包调度方法来为用户提供可靠的端对端服务质量（QOS）保证。

目前一类基于 GPS 模型的调度算法如 WFQ、$WF^2Q$、$WF^2Q$+等得到了广泛的研究。GPS (Generalized Processor Sharing)模型是一种理想化的流体模型，它基于如下两个假设：其一，数据包长度无限可分；其二，所有的流可以同时接受服务。而在实际系统中，一个调度器服务的最小单元为一个数据包，而且一次只能为一个流服务，因此 GPS 模型是不可实现的。于是 Jon C. R. Bennett 和 Hui Zhang（见文献 J. Bennett and H. Zhang, Hierarchical packet fair queuing algorithms, *In Proceedings of the ACM-SIGCOMM'96*, pages 143-156, Palo Alto, CA, August 1996）提出了 $WF^2Q$+调度算法用于在实际系统中模拟 GPS 模型。其基本思想是，对流中的每个数据包维护一个开始服务时间和结束服务时间，当调度器准备发送一个数据包时，首先需要对等待调度的数据包进行资格测试，只有开始服务时间小于系统的虚拟时间的数据包才可以通过测试，然后在通过测试的数据包中选择具有最小结束服务时间的数据包进行发送，这种策略称为 SEFF (Smallest Eligible virtual Finish time First)选择策略。

由于 $WF^2Q+$具有很好的公平性和时延性，而且算法复杂度比较低，因此得到了业界的广泛关注。但是，$WF^2Q+$算法在应用上存在如下问题：1）算法的复杂度随着受调度流的数量增加而增大，特别是在高速核心路由器上，当数据流的数量很大时，算法的应用将会给系统带来巨大的负担，2）在硬件上实现很困难。

**发明内容**

本发明的目的在于提供一种能克服上述缺陷的数据包的调度方法，该方法简单、高效，便于硬件实现，并基本保证了 $WF^2Q+$算法的性能。

为实现上述发明目的，一种数据包的调度方法，将待调度的数据包分成新到流队列和积压流队列两部分，由调度器进行调度，其调度步骤为：

(1) 初始化调度节点，设定系统虚拟时间的初始值；

(2) 当某个数据包到达调度节点时，检查这个数据包是否为其所属数据流的第一个数据包，如果是，根据其速率和/或长度挂到新到流队列部分相应的 $Q_1$ 队列的尾部，并通过公式(3)，计算这个数据包的虚起始服务时间，该虚起始服务时间即该数据流的虚起始服务时间；如果这个数据包不是这个数据流的第一个数据包，直接将这个数据包挂在所属数据流的尾部；

(3) 进行调度时，调度器会扫描所有队列中第一个数据流的头数据包的虚起始服务时间，从中找出虚起始服务时间小于系统虚拟时间的合法数据包，再通过公式(4)，计算这些合法数据包的虚结束服务时间，找出具有最小虚结束服务时间的数据包进行发送；

(4) 对选定的头数据包进行发送的过程为：首先将该数据包从所属的数据流 F 中取下来发送，数据流 F 再根据其新的头数据包的速率和长度，挂在积压流队列部分相应的 $Q_2(R_x, L_y)$ 队列的尾部，通过公式(2)更新数据流 F 的虚起始服务时间，即数据流 F 新的

头数据包的虚起始服务时间，并通过公式(1)更新系统虚拟时间；

$$V(t+\tau)=\max\left(V(t)+\tau,\min_{i\in B(t)}\left(S_i^{h_i(t)}\right)\right)\cdots\cdots(1)$$

$$S_i^k=\begin{cases}F_i^{k-1}\rightarrow Q_i\neq 0\cdots\cdots(2)\\ \max\left(V\left(a_i^k\right),F_i^{k-1}\right)\rightarrow Q_i=0\cdots\cdots(3)\end{cases}$$

$$F_i^k=S_i^k+\frac{L_i^k}{R_i(t)}\cdots\cdots(4)$$

(5) 重复步骤(2)至(4)，直至工作完成。

为了验证本调度方法的性能，在 Network Simulator 仿真环境中，实现了本调度方法，并检查了如下性能指标：

- ✧ 带宽：　　每个数据流得到的实际带宽(Mbps)
- ✧ 带宽抖动：每个数据流在相邻时间段内得到的平均实际带宽之差(Mbps)
- ✧ 时延：　　每个数据流的数据包离开与到达的时间差(ms)
- ✧ 时延抖动：每个数据流前后两个数据包的平均时延差(ms)

其结果见具体实施方式部分。总之，本调度方法简单高效，易于硬件实现，并基本保证了 $WF^2Q+$算法原有的优良性能。

**附图说明**

图 1 为本发明调度方法的示意图；
图 2 为调度方法的实现框架；
图 3 为简化调度方法的实现框架；
图 4 为仿真拓扑图；
图 5 为使用调度方法时数据流 1 的带宽特性；
图 6 为使用调度方法时数据流 1 的带宽抖动特性；

图 7 为使用调度方法时数据流 1 的时延特性；
图 8 为使用调度方法时数据流 1 的时延抖动特性；
图 9 为使用简化调度方法时数据流 1 的带宽特性；
图 10 为使用简化调度方法时数据流 1 的带宽抖动特性；
图 11 为使用简化调度方法时数据流 1 的时延特性；
图 12 为使用简化调度方法时数据流 1 的时延抖动特性。

**具体实施方式**

为了便于下面的描述，先定义一些符号说明和公式：

| 符号 | 定义 |
|---|---|
| $V(t)$ | 系统虚拟时间函数 |
| $S_i^k$ | 第 i 个数据流的第 k 个数据包的虚起始服务时间 |
| $F_i^k$ | 第 i 个数据流的第 k 个数据包的虚结束服务时间 |
| $\tau$ | 系统虚拟时间更新的时间间隔 |
| $B(t)$ | t 时刻系统中等待调度的所有流的集合 |
| $h_i(t)$ | t 时刻第 i 个数据流的头数据包 |
| $Q_i$ | 第 i 个数据流当前待调度的数据包个数 |
| $a_i^k$ | 第 i 个数据流的第 k 个数据包的到达时间 |
| $L_i^k$ | 第 i 个数据流的第 k 个数据包的长度 |
| $R_i(t)$ | t 时刻第 i 个数据流的速率 |

$$V(t+\tau)=\max\left(V(t)+\tau,\min_{i\in B(t)}\left(S_i^{h_i(t)}\right)\right)\cdots\cdots(1)$$

$$S_i^k=\begin{cases}F_i^{k-1}\rightarrow Q_i\neq 0\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots(2)\\ \max\left(V\left(a_i^k\right),F_i^{k-1}\right)\rightarrow Q_i=0\cdots\cdots\cdots\cdots(3)\end{cases}$$

$$F_i^k=S_i^k+\frac{L_i^k}{R_i(t)}\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots(4)$$

下面结合附图对本发明进行详细的描述：

在本发明所述调度方法中，针对不同的数据流，采用速率对其进行量化，其量化等级数为 M，依次记为 $R_1$, ..., $R_M$。针对数据流中的不同数据包，采用长度对其进行量化，其量化等级数为 N，依次记为 $L_1$, ..., $L_N$（对于不同的数据流，其长度量化等级数可以不同，也可以相同，为了简化实现，采用相同的长度量化等级数）。R 与 L 的组合可以得到不同的队列，每个队列记为 $Q(R_m, L_n)$，这样就共有 M×N 个队列。

每个数据流按照其预留的速率 R（如 $R=R_m$）和头数据包长度 L（如 $L_{n-1}<L\leq L_n$），挂在相应的队列 $Q(R_m, L_n)$上。头数据包是指数据流中当前排在最前面待调度的数据包，其长度量化为比它大的最近等级长度；大于最高等级长度的，量化为最高等级长度。队列上的数据流依次记为 $F_1$, $F_2$, ...，最后一个数据流记为 $F_{Tail}$。数据流 $F_i$ 中的数据包依次记为 $P_{i1}$，$P_{i2}$，...，最后一个数据包记为 $P_{i,Tail}$。

数据流第一个数据包是指该数据流初次发生或停止一段时间后重新发生时最先被处理的数据包。确认一个新到的数据包为其所属数据流第一个数据包的依据是该数据包所属数据流当前没有数据包正在等待调度，如果该数据包所属数据流已有数据包正在等待调度，则称该数据包为其所属数据流的后续数据包。由于数据流第一个数据包和其后续数据包的虚起始服务时间的计算不一致，需要对该数据流的第一个数据包单独处理。于是我们将队列分成两个部分，新到流队列部分①和积压流

队列部分②，如图 1 所示。新到流队列部分①负责处理数据流的第一个数据包，积压流队列部分②负责处理数据流的后续数据包，两部分共需 2×M×N 个队列。调度器③的作用是采用 SEFF 策略对这两部分队列进行统一调度。

结合图 2，本调度方法的具体调度步骤如下：

(1) 初始化调度节点，设定系统虚拟时间的初始值，比如 0；

(2) 当某个数据包到达调度节点时，检查这个数据包是否为其所属数据流的第一个数据包，如果是，根据其速率和长度挂到新到流队列部分①相应的 $Q_1(R_m, L_n)$队列的尾部，并通过公式(3)，计算这个数据包的虚起始服务时间，该虚起始服务时间也是该数据流的虚起始服务时间。如果这个数据包不是这个数据流的第一个数据包，直接将这个数据包挂在所属数据流的尾部；

(3) 进行调度时，调度器③会扫描所有队列（包括 M×N 个新到流队列 $Q_1(R, L)$和 M×N 个积压流队列 $Q_2(R, L)$）中第一个数据流的头数据包的虚起始服务时间，从中找出虚起始服务时间小于系统虚拟时间的合法数据包，再通过公式(4)，计算这些合法数据包的虚结束服务时间，找出具有最小虚结束服务时间的数据包进行发送；

(4) 无论是选定新到流队列 $Q_1(R_m, L_n)$的第一个数据流的头数据包进行发送，还是选定积压流队列 $Q_2(R_m, L_n)$的第一个数据流的头数据包进行发送，均做出这样的处理：首先将该数据包从所属的数据流 F 中取下来发送，数据流 F 再根据其新的头数据包的速率和长度，挂在积压流队列部分②相应的 $Q_2(R_x, L_y)$队列的尾部，通过公式(2)更新数据流 F 的虚起始服务时间，即数据流 F 新的头数据包的虚起始服务时间，并通过公式(1)更新系统虚拟时间；

(5) 重复步骤(2)至(4)，直至工作完成。

本发明所指的调度节点，即应用了本调度方法的设备，如路由器。

为了减少队列数目，节省硬件资源，我们同时提出了简化调度方法。如图 3 所示，对于新到流队列部分①只采用长度（或速率）进行量化，记为 $Q_1$（$L_n$）（或 $Q_1$（$R_m$）），这样共需要 M×N+N（或 M×N+M）个队列。以下将以长度量化为例。

结合图 3，简化调度方法的具体调度步骤如下：

(1) 初始化调度节点，设定系统虚拟时间的初始值，比如 0；

(2) 当某个数据包到达调度节点时，检查这个数据包是否为其所属数据流的第一个数据包，如果是，根据其长度挂到新到流队列部分①相应的 $Q_1(L_n)$队列的尾部，并通过公式(3)，计算这个数据包的虚起始服务时间，该虚起始服务时间也是该数据流的虚起始服务时间。如果这个数据包不是这个数据流的第一个数据包，直接将这个数据包挂在所属数据流的尾部。

(3) 进行调度时，调度器③会扫描所有队列（包括 N 个新到流队列 $Q_1(L)$和 M×N 个积压流队列 $Q_2(R, L)$）中第一个数据流的头数据包的虚起始服务时间，从中找出虚起始服务时间小于系统虚拟时间的合法数据包，再通过公式(4)，计算这些合法数据包的虚结束服务时间，找出具有最小虚结束服务时间的数据包进行发送。

(4) 无论是选定新到流队列 $Q_1(L_n)$的第一个数据流的头数据包进行发送，还是选定积压流队列 $Q_2(R_m, L_n)$的第一个数据流的头数据包进行发送，均做出这样的处理：首先将该数据包从所属的数据流 F 中取下来发送，数据流 F 再根据其新的头数据包的速率和长度，挂在积压流队列部分②相应的 $Q_2(R_x, L_y)$队列的尾部，通过公式(2)更新数据流 F 的虚起始服务时间，即数据流 F 新的头数据包的虚起始服务时间，并通过公式(1)更新系统虚拟时间；

(5) 重复步骤(2)至(4)，直至工作完成。

具体的仿真拓扑结构如图 4 所示。

在这个仿真拓扑结构中，各输入链路和输出链路带宽均为 10M，所有的数据流均受到调度节点的调度并从输出链路输出。在图 4 的调度节点，调度方法和简化调度方法的速率和长度均被量化成 5 个等级。

对数据流定义 5 个速率等级：

- ✧ 0.1Mbps.
- ✧ 0.3Mbps.
- ✧ 1 Mbps.
- ✧ 2 Mbps.
- ✧ 5 Mbps.

对数据包定义 5 个长度等级：

- ✧ 200 bytes.
- ✧ 400 bytes.
- ✧ 800 bytes.
- ✧ 1000 bytes.
- ✧ 1600 bytes.

对数据流定义两种传输方式：

- ✧ CBR（Const Bit Rate）：表明数据流以恒定的速率传输；
- ✧ On/Off：表明数据流以某个速率间隙性地传输。

对数据包长度定义三种分布方式：

- ✧ 相同长度：数据流中的所有数据包长度相同；
- ✧ 平均分布：数据流中的数据包长度在某个长度范围内均匀分布；
- ✧ 正态分布：数据流中的数据包长度以某个长度为中心正态分布。

对两种调度方法进行了大量测试，选取一个较典型的测试配置以说明测试结果。调度方法的典型仿真参数如下表所示：

| 数据流标识 | 链路带宽 | 预约带宽 | | 数据流传输方式和速率 | | 数据包长度 |
|---|---|---|---|---|---|---|
| 1 | 10 Mbps | 5.0 Mbps | 50 % | On/Off | 5.0 Mbps | 正态分布 |
| 2 | 10 Mbps | 2.0 Mbps | 20 % | On/Off | 2.0 Mbps | 正态分布 |
| 3 | 10 Mbps | 1.0 Mbps | 10 % | On/Off | 1.0 Mbps | 正态分布 |
| 4 | 10 Mbps | 1.0 Mbps | 10 % | On/Off | 1.0 Mbps | 正态分布 |
| 5 | 10 Mbps | 0.3 Mbps | 3 % | On/Off | 0.3 Mbps | 正态分布 |
| 6 | 10 Mbps | 0.3 Mbps | 3 % | On/Off | 0.3 Mbps | 正态分布 |
| 7 | 10 Mbps | 0.1 Mbps | 1 % | On/Off | 0.1 Mbps | 正态分布 |
| 8 | 10 Mbps | 0.1 Mbps | 1 % | On/Off | 0.1 Mbps | 正态分布 |
| 9 | 10 Mbps | 0.1 Mbps | 1 % | On/Off | 0.1 Mbps | 正态分布 |
| 10 | 10 Mbps | 0.1 Mbps | 1 % | On/Off | 0.1 Mbps | 正态分布 |
| 调度出口 | 10 Mbps | | | | | |

上述配置中每个数据流均采用 On/Off 模型，数据流中的数据包长度采用正态分布（均值为 1000 字节，方差 400 字节）。对调度方法和简化调度方法进行仿真实验，得到数据流 1 在两种方法中的性能指标：带宽、带宽抖动、时延和时延抖动。如图 5～图 12 所示。

根据图 5～图 12 所示的仿真结果，可以看出：这两种调度方法均能较好地保证数据流 1 的带宽、带宽抖动、时延和时延抖动等性能，从而能较好地保证用户的服务质量。

在 NS 仿真实验的基础上，采用 Xilinx 公司的 FPGA 实现了支持简化调度方法的调度芯片。该芯片最大支持 128k 个数据流，5 个速率等级和 5 个长度等级，并可动态配置各速率和长度等级的值。经实际运行测试，该芯片能很好地保证各数据流的预约带宽、时延性及公平性。

# 说明书附图

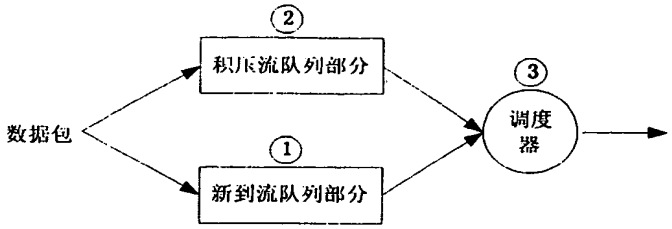



图 1

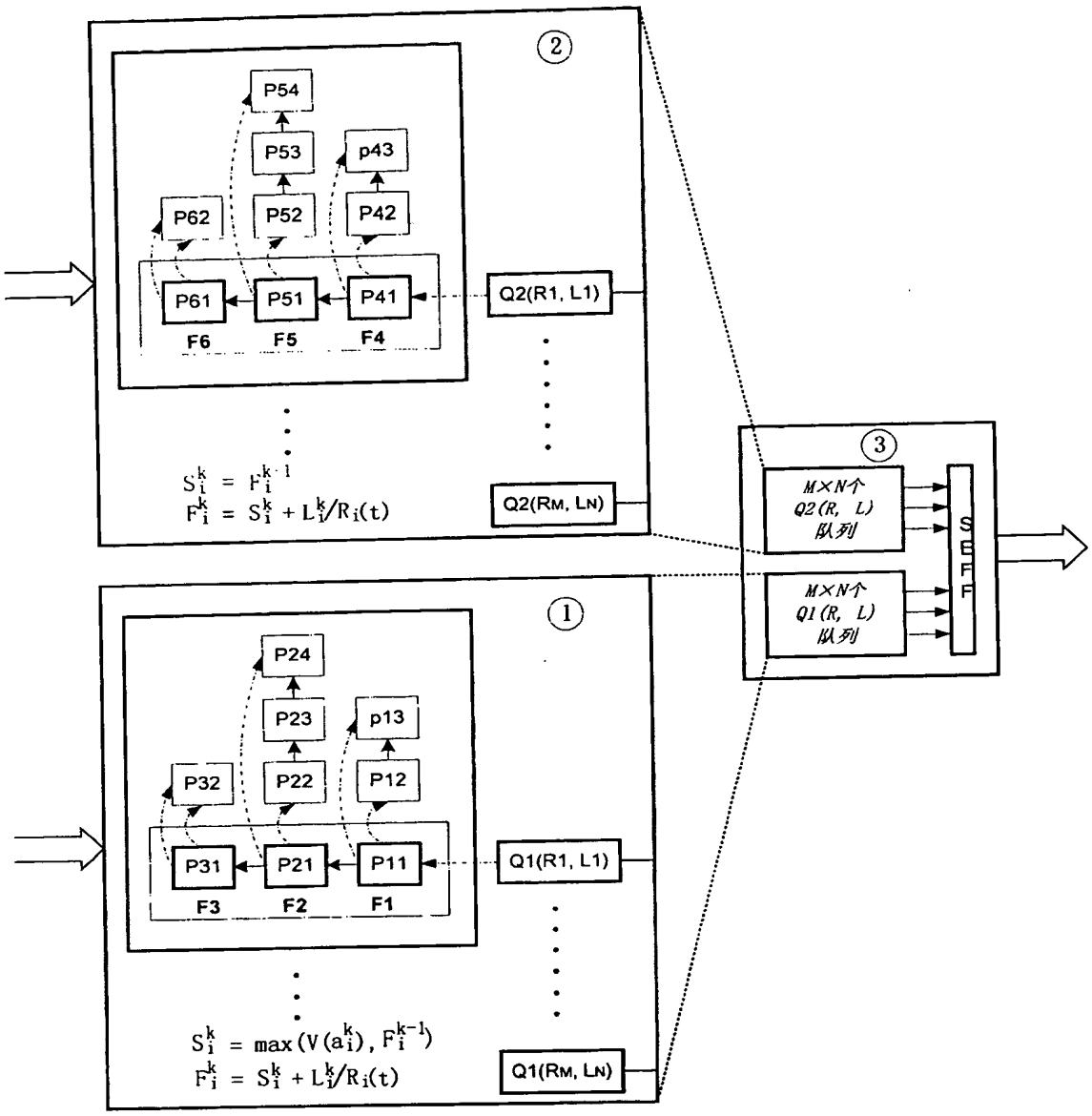

②
P54
P53
p43
P62
P52
P42
P61
P51
P41
F6
F5
F4
Q2(R1, L1)
Q2(RM, LN)
$S_i^k = F_i^{k-1}$
$F_i^k = S_i^k + L_i^k/R_i(t)$
③
M×N个
Q2(R, L)
队列
M×N个
Q1(R, L)
队列
SCFQ
①
P24
P23
p13
P32
P22
P12
P31
P21
P11
F3
F2
F1
Q1(R1, L1)
Q1(RM, LN)
$S_i^k = \max(V(a_i^k), F_i^{k-1})$
$F_i^k = S_i^k + L_i^k/R_i(t)$

图 2

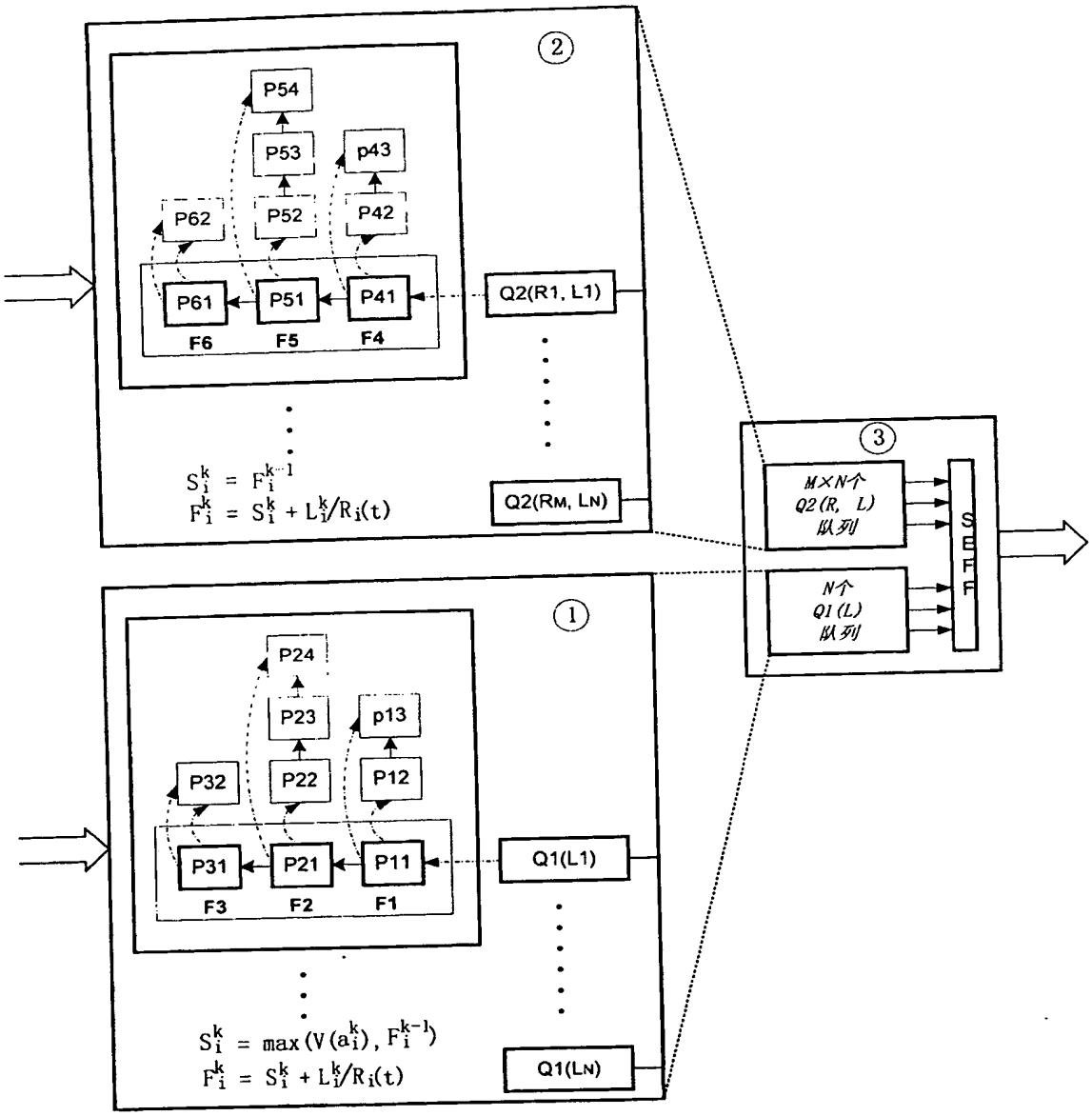

②
P54
P53
p43
P62
P52
P42
P61
P51
P41
F6
F5
F4
Q2(R1, L1)
Q2(RM, LN)
$S_i^k = F_i^{k-1}$
$F_i^k = S_i^k + L_i^k/R_i(t)$
③
M×N个
Q2(R, L)
队列
N个
Q1(L)
队列
SELECT
①
P24
P23
p13
P32
P22
P12
P31
P21
P11
F3
F2
F1
Q1(L1)
Q1(LN)
$S_i^k = \max(V(a_i^k), F_i^{k-1})$
$F_i^k = S_i^k + L_i^k/R_i(t)$

图 3

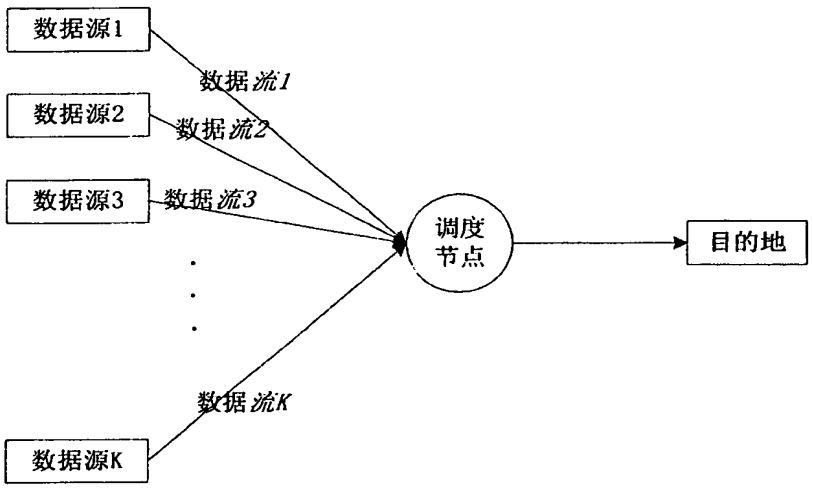



图 4

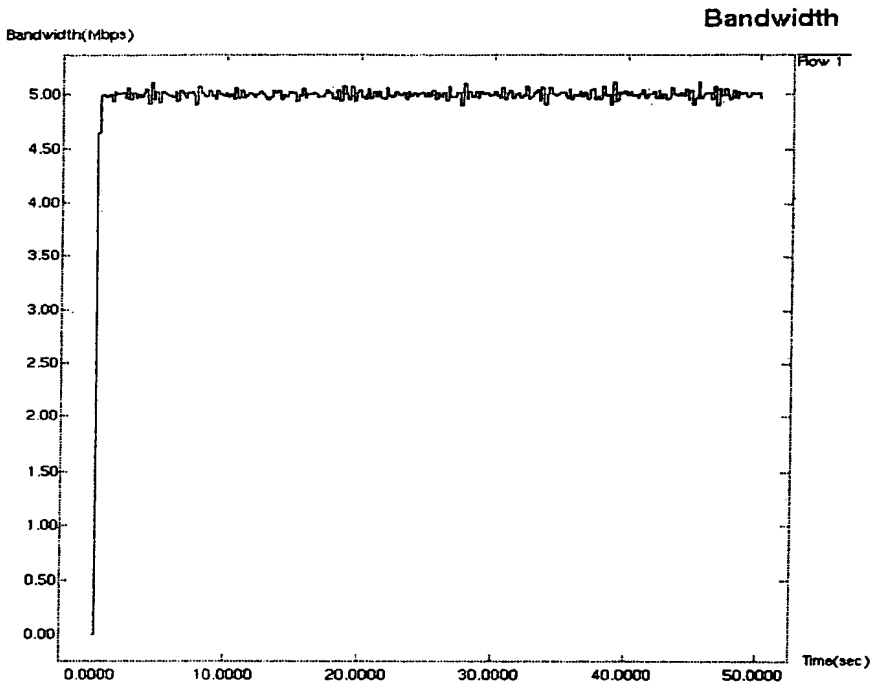

Bandwidth
Bandwidth(Mbps)
Flow 1
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0.0000
10.0000
20.0000
30.0000
40.0000
50.0000
Time(sec)

图 5

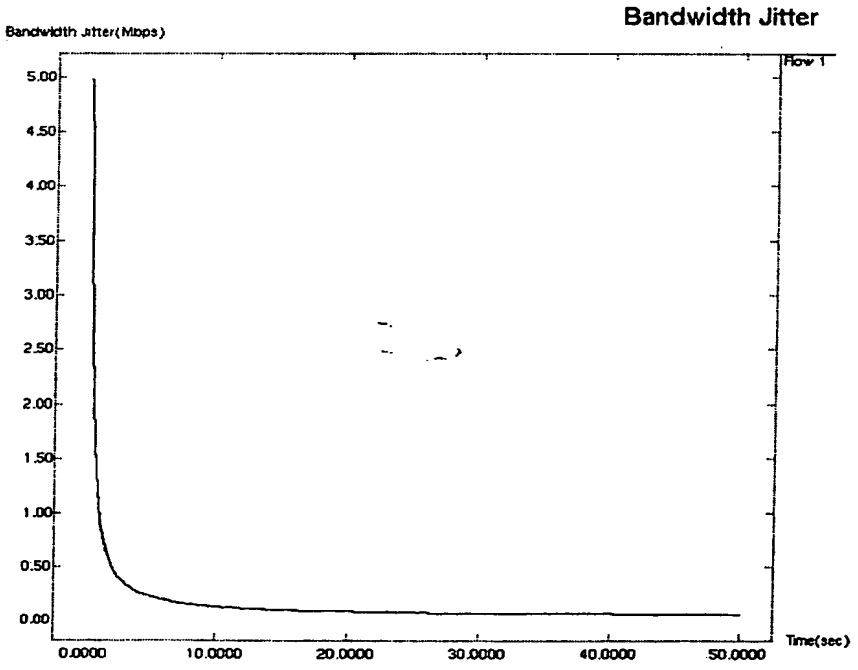

Bandwidth Jitter
Bandwidth Jitter(Mbps)
Flow 1
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0.0000
10.0000
20.0000
30.0000
40.0000
50.0000
Time(sec)

图 6

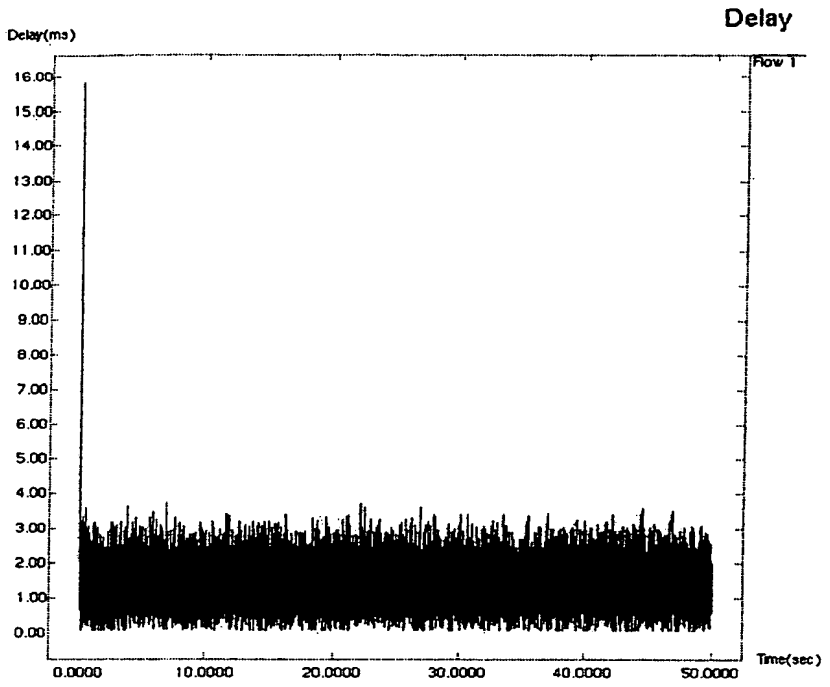

Delay
Delay(ms)
Flow 1
16.00
15.00
14.00
13.00
12.00
11.00
10.00
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
0.0000
10.0000
20.0000
30.0000
40.0000
50.0000
Time(sec)

图 7

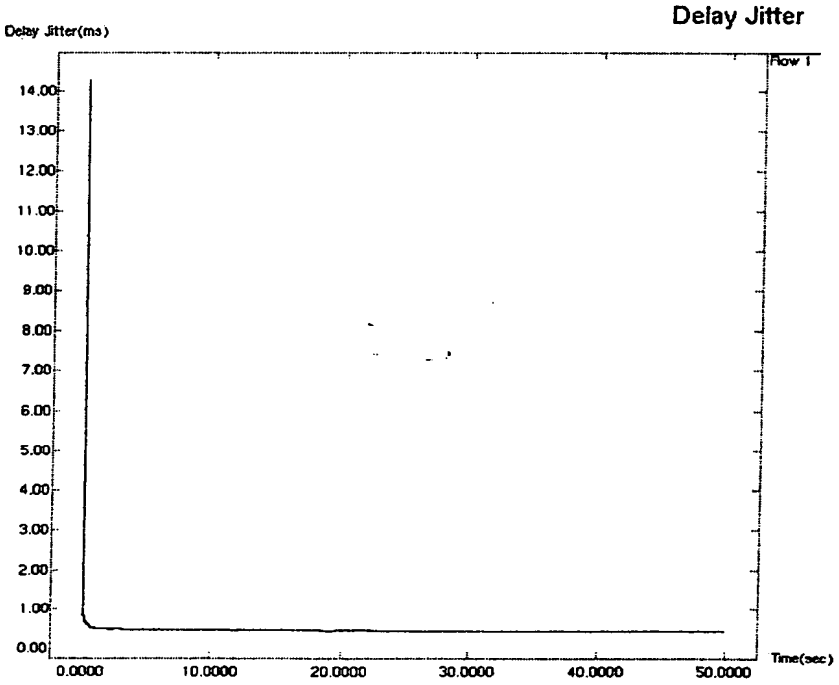

Delay Jitter
Delay Jitter(ms)
Flow 1
14.00
13.00
12.00
11.00
10.00
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
0.0000
10.0000
20.0000
30.0000
40.0000
50.0000
Time(sec)

图 8

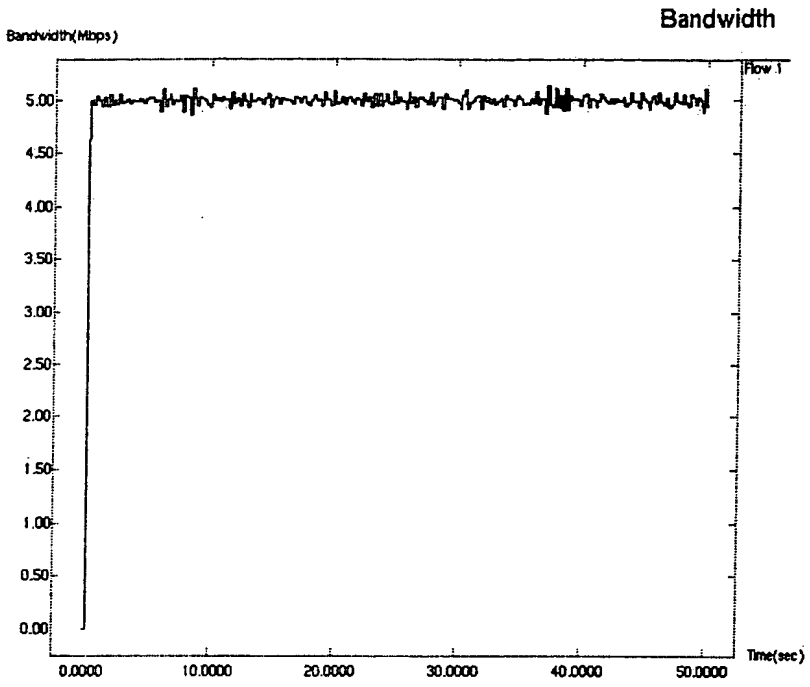

Bandwidth
Bandwidth(Mbps)
Flow 1
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0.0000
10.0000
20.0000
30.0000
40.0000
50.0000
Time(sec)

图 9

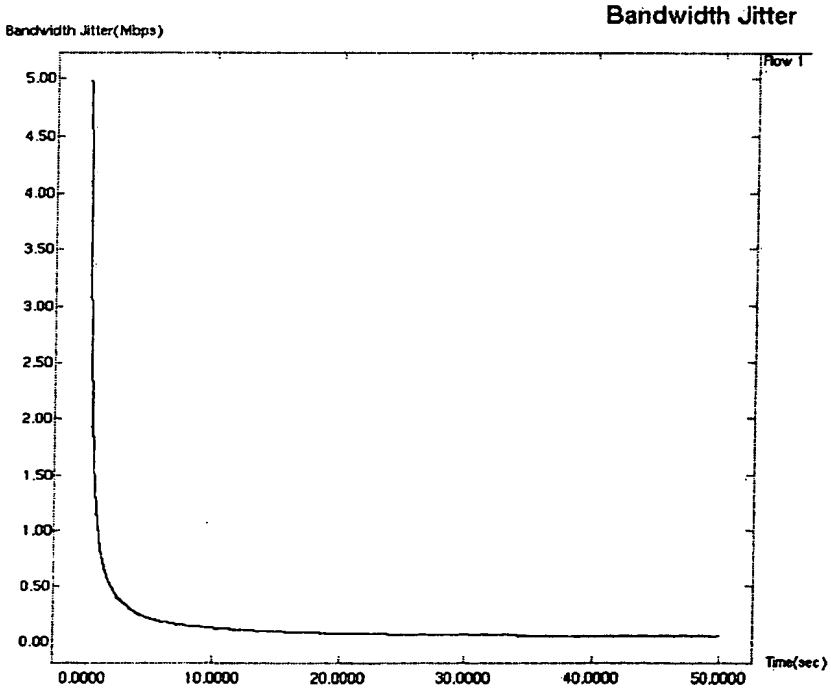

Bandwidth Jitter
Bandwidth Jitter(Mbps)
Flow 1
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0.0000
10.0000
20.0000
30.0000
40.0000
50.0000
Time(sec)

图 10

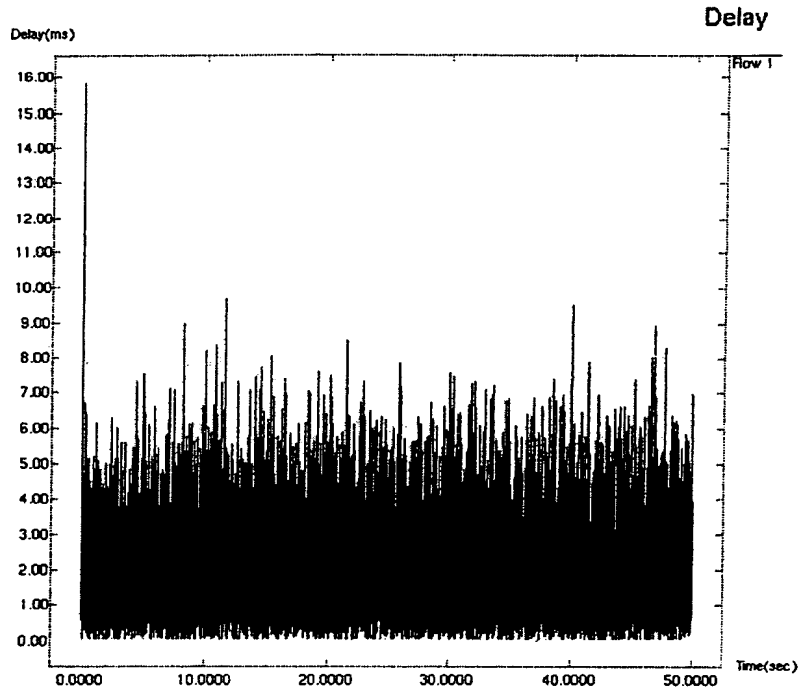

Delay
Delay(ms)
Flow 1
16.00
15.00
14.00
13.00
12.00
11.00
10.00
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
0.0000
10.0000
20.0000
30.0000
40.0000
50.0000
Time(sec)

图 11

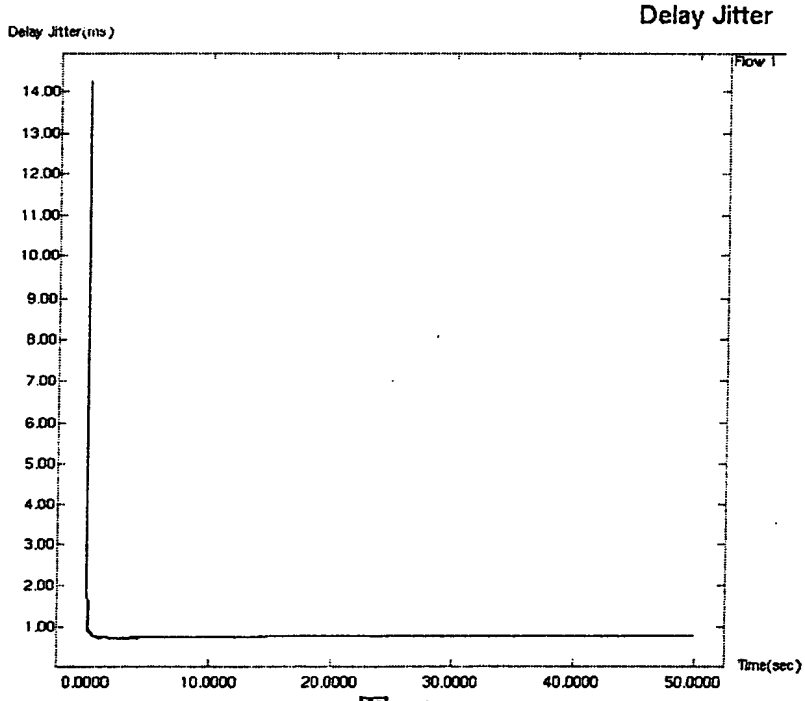

Delay Jitter
Delay Jitter(ms)
Flow 1
14.00
13.00
12.00
11.00
10.00
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.0000
10.0000
20.0000
30.0000
40.0000
50.0000
Time(sec)

图 12